



**Isomorphic**  
SOFTWARE

---

# **Messaging Quick Reference**

## **SmartClient™ Real-Time Messaging**

SmartClient v6.5  
April 2008

***Messaging Quick Reference***  
***SmartClient™ Real-Time Messaging***  
SmartClient v6.5

Copyright ©2001-2008 Isomorphic Software, Inc. All rights reserved. The information and technical data contained herein are licensed only pursuant to a license agreement that contains use, duplication, disclosure and other restrictions; accordingly, it is "Unpublished-rights reserved under the copyright laws of the United States" for purposes of the FARs.

**Isomorphic Software, Inc.**  
**109 Stevenson Street, Level 4**  
**San Francisco, CA 94105**  
**U.S.A.**

**Web:**      [www.isomorphic.com](http://www.isomorphic.com)  
**Email:**    [info@isomorphic.com](mailto:info@isomorphic.com)  
              [support@smartclient.com](mailto:support@smartclient.com)  
              [feedback@smartclient.com](mailto:feedback@smartclient.com)

**Notice of Proprietary Rights**

The software and documentation are copyrighted by and proprietary to Isomorphic Software, Inc. ("Isomorphic"). Isomorphic retains title and ownership of all copies of the software and documentation. Except as expressly licensed by Isomorphic in writing, you may not use, copy, disseminate, distribute, modify, reverse engineer, unobfuscate, sell, lease, sublicense, rent, give, lend, or in any way transfer, by any means or in any medium, the software or this documentation.

1. These documents may be used for informational purposes only.
2. Any copy of this document or portion thereof must include the copyright notice.
3. Commercial reproduction of any kind is prohibited without the express written consent of Isomorphic.
4. No part of this publication may be stored in a database or retrieval system without prior written consent of Isomorphic.

**Trademarks and Service Marks**

Isomorphic Software, SmartClient, and all Isomorphic-based trademarks and logos that appear herein are trademarks or registered trademarks of Isomorphic Software, Inc. All other product or company names that appear herein may be claimed as trademarks or registered trademarks of their respective owners.

**Disclaimer of Warranties**

THE INFORMATION CONTAINED HEREIN IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT AND ONLY TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

## Description

The optional Real-Time Messaging (RTM) module for SmartClient allows browser-based web applications to:

- publish and subscribe to HTTP messaging channels
- send and receive messages via server push, for “real-time” updates without polling

This functionality is currently supported for deployment on Java server platforms only.

## Example

For a simple example of publish and subscribe messaging, see:

`<yourhost>/isomorphicSDK/examples/messaging/simple_chat.jsp`

You must install the SmartClient server components to run this example.

## Client API summary

Client-side RTM interfaces are provided by the SmartClient *Messaging* service:

---

```
Messaging.subscribe(channel, callback)
```

Subscribes the client to the messaging channel identified by *channel*. When the server sends a message on this channel, the *callback* function or script will be called on the client with a single *data* parameter.

---

```
Messaging.unsubscribe(channel)
```

Unsubscribes the client from the messaging channel identified by *channel*.

---

```
Messaging.getSubscribedChannels()
```

Returns an array of identifiers for all currently subscribed channels on this client.

---

```
Messaging.send(channels, data)
```

Send any data (primitive type, or compound data structure) to a channel or list of channels. *channels* may be a single channel identifier (string) or an array of channel identifiers. *data* will be passed to the callback specified in any `subscribe()` call for a matching channel.

## Server API summary

Server-side RTM interfaces are provided by the following classes in `com.isomorphic.messaging`:

---

### **ISCMessagesDispatcher**

Abstract class; use to obtain a concrete message dispatcher class.

---

#### **instance()**

returns a concrete ISCMessagesDispatcher class capable of delivering responses within the JVM

---

#### **instance(RequestContext context)**

returns a concrete ISCMessagesDispatcher class capable of delivering responses to web browsers

---

#### **send(ISCMessages msg)**

send a message

---

#### **register(ISubscriber subscriber)**

register this subscriber

---

#### **unregister(ISubscriber subscriber)**

unregister this subscriber

---

#### **subscribe(ISubscriber subscriber, String channel)**

subscribe a given subscriber to a given channel

---

#### **unsubscribe(ISubscriber subscriber, String channel)**

unsubscribe a given subscriber from a given channel

---

#### **boolean isSubscribed(ISubscriber subscriber, String channel)**

check to see if a given subscriber is subscribed to a given channel

---

**ISCMMessage**

Message object.

---

```
ISCMMessage(String channel, Object data)
```

```
ISCMMessage(List channels, Object data)
```

create a message bound for the specified channel(s) with the specified data payload

---

**ISubscriber**

Simple interface for a message subscriber. Required methods are:

---

```
public void send(ISCMMessage msg) throws Exception;
```

---

```
public ISCMMessage nextMessage(long timeout) throws  
Exception;
```

---

**ISCSubscriber**

Simple concrete implementation of ISubscriber. `send()` adds message to a queue and `nextMessage()` retrieves them.

## Configuration

The SmartClient message dispatcher can operate in *simple* mode or *enterprise* mode:

- Simple mode uses an in-memory messaging delivery system with no message persistence, and can operate only in the context of a single JVM.
- Enterprise mode uses Java Message Service (JMS) as the messaging backend, and can operate in a clustered environment.

You can set the following properties in `WEB-INF/classes/server.properties` to control this mode and other parameters:

---

```
# how often do we send keepalives to the client (ms)

messaging.keepaliveInterval: 3000

# how long the client waits after the keepaliveInterval before re-establishing
# the connection (ms)

messaging.keepaliveReestablishDelay: 1000

# how long the client waits for the connect handshake to complete before
# retrying

messaging.connectTimeout: 4000

# connection time to live - the maximum amount of time a persistent connection
# is allowed to stay open before being re-established (ms)

messaging.connectionTTL: 120000

# total response size to pad out to in order to defeat intervening
# buffering by proxies (bytes)

messaging.flushBufferSize: 8096

# dispatcher to use for user registration/message queueing
# com.isomorphic.messaging.LocalMessageDispatcher for simple one-jvm messaging
# com.isomorphic.messaging.JMSMessageDispatcher for JMS-backed messaging

messaging.dispatcherImplementer:
    com.isomorphic.messaging.LocalMessageDispatcher

# jms configuration - for JMSMessageDispatcher only

messaging.jms.context: _container_
messaging.jms.jndiPrefix: jms
messaging.jms.topicConnectionFactory: TopicConnectionFactory
```

---



## Tips

- We recommend developing using a single container with the LocalMessageDispatcher and then switching to the JMSMessageDispatcher for production.
- For best performance, we recommend forwarding the messaging connections to an HTTP 1.0 server. This results in better interactive behaviour in Internet Explorer web browsers.